

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <04/16/23> | <Scott Bosdash> | Recommended sections altered |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room would like to create a game that is web-based and runs on multiple platforms by the name of “Draw It or Lose It.” This game is currently only available for use on the android platform. The game will consist of teams of several people and will last for four rounds with one minute intervals. A picture will be pulled from a library, then the current team will guess what it is until time runs out. Assuming the time runs out without the team answering, each team member from the opposing team will get a chance to answer for fifteen seconds.

## Requirements

The requirements that need to be followed while writing the code and the software is only one part of this game. The application development is another part altogether. Since the Gaming Room would like this game to run on all devices, we will need to change it from only being on Android to be able to run on machines with Windows, Linux, or iOS. For this to happen, we will need to rewrite code in swift for Apple devices, or plan out a way that the existing code can be recycled through inheritance of other languages. Using multiple languages to create a stronger code would be optimal.

## [Design Constraints](#_2et92p0)

There will need to be multiple teams involved in the game, each team will have several people on it. Each game and team have to be unique so that users can see if a name is currently in use. One instance of the game may exist at any given time, and the game must run on multiple platforms.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_8h2ehzxfam4o)

Entity will create a relationship between the game, team, and the player class, which means that they all will inherit information from entity. By using UML, we can show the inheritance. All of the classes will have a shared reference such as “name” and “id,” which is why entity is a superclass. This means that “team” and “player” will have a “has a” type, and Game has a team and GameService has Games. When using UML this “has a” relationship is known as “aggregation.” The term “has a” means that it is an instance of one class with a reference to the instance of another. If you look at the diagram, GameService has a reference of Games, which is a reference of Team, and Team is a reference of Player.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac used to have a server platform, but has since been discontinued as of April 2022, however it still can be used as a server. It has higher security, lower virus threat than windows. Neat and clean setup, high support from customer service, can also use NTFS and FAT formatted hard drives. There can be server-based deployment for Mac OS in High Sierra and anything newer. | Linux can be difficult, but it is open source and free to use by anyone willing to learn and use it. Nearly impossible to get a virus without actively trying. It is the best choice for a server due to the reasons stated, and usually outperforms other OS by a huge margin. There can be server-based deployment for Linux for free, just needs to be installed. | In Windows, it is common to get viruses, fairly expensive. customer service is available but often poor quality. Although, Windows is improving as of late, with hyper v technology, virtualization and being linked to Azure. A good contender overall for a server with Windows 10 or 11. There can be server-based deployment for Windows with Windows Server 2019 and newer most easily, but there are many paid and free option. | For Android, there is a tool that you may use called “KSWEB” that turns your android device into a portable web server for 3 dollars. Add in some information into the app and set your server up. It is convenient for testing purposes, but ultimately not worth using over another, much better option. There can be server-based deployment on Android by using Android enterprise, a low cost tool, there are probably free alternatives however. |
| **Client Side** | Less used than Windows, price is similar to Windows. Has a GUI like Windows making it equivalent, some games may not work, but most do. Comes standard if purchased from a retailer. No added costs for the user. | Best security for the user, also secure on both sides, not as much front-end interaction as another OS. Perhaps the best choice. No added cost for user. | Most people have windows devices, security isn’t great, but better than Android. Easy to use for most people with an app installation. Comes standard if PC is purchased from a retailer. | Processing power would be limited almost exclusively to the server if an app is intended to function properly, also, security threats would be higher. Different versions may interact differently, depending on device purchased, OS must be updated, depending on app. |
| **Development Tools** | Mac OS tools – There are an abundance of tools to use for Mac OS, some of the best would be: X Code which is great all around and can also be used multiplatform, Visual Studio Code, which is my own, preferred IDE to use, and Postman which also has an API repository, making it unique to some of the other options. The IDEs make life as easy as possible, and function according to the user’s needs, whether it is on team or many teams. | Linux tools - Again, many great tools, but to narrow it down to a few of the best: Gedit is a simple yet effective tool, Vim is another great tool for coding, but can also be used to edit config files and even xml docs and comes standard in most Linux distros, Netbeans is a popular option, partly because it is multiplatform. IDE tools don’t vary a whole lot while using them, they work as intended, so learning them is the only wall. One team or many should not make a difference. | Windows tools – Too many options, but not all are great, being productive is the goal, so I will list some of the best: VSC, by far, in my opinion at least, the best development tools, uses multiple languages and libraries, color coded for readability, all around, near perfect. Xcode is great, but unavailable for use on Android, Firefox developer tools is good for CSS, JavaScript, and HTML editing. Windows is the most used system, so learning to use the tools should be as common overall. One team or many, the tools are capable of handling either event. | Android –  Somewhat limited options, or at least “good options.”  Android studio is one of the best and most used from what I can tell, it can use AI and augmented reality. ADB has been around forever, and still works well for most applications but requires a PC to connect to, Genymotion will allow a dev to test their products and can also debug, could be useful for testing purposes, also uses drag and drop features to make things easier. Android is becoming mor common, thus more people are learning to use them effectively, solo or in a team, or multiple teams. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: I would recommend using a Linux server using Ubuntu. Since it is one of the most well-known distributions available and works quite well for all applications on all platforms. It may seem a little unwieldy, but it is one of the easier Linux servers to use and to master. Windows is okay, but Linux is far better, to give the user and the Gaming Room the best experience possible, that is my recommendation.
2. **Operating Systems Architectures**: Since we will be using Linux Ubuntu as the server, the operating system architecture will be the most stable. It looks vastly different from Windows, but if someone who has worked with it previously operates it, there should be no issues. You have a choice when using Ubuntu to enhance your desktop environment, making it customizable and giving the system a more user-friendly feel. Ubuntu is also a great resource for cloud computing.
3. **Storage Management**: Storage management is something that makes a bit of a difference, you have a lot of options. Some of the options are standard hard drives, solid state drives, M2 solid state drives, or even cloud storage. I would recommend using SSD (Solid State Drive), it is much faster than a standard HDD (hard disk drive) and less costly than M2 drives or cloud storage. Cloud storage is useful, but you must pay monthly or yearly for it, making the other more fiscally irresponsible in my opinion. The initial investment in a SSD setup would overall be a better option.
4. **Memory Management**: Memory management for the server can be done with Azure marketplace, it is a great resource for any company. It can optimize memory usage during peak times and has a lot of other helpful tools built into it. It was developed by Microsoft, but it can connect to any device and on any platform, it works in real-time 24 hours a day and has great support if you were to need it. You may also use VM (Virtual Machines) while using Ubuntu, making it even more useful for the Gaming Room.
5. **Distributed Systems and Networks**: Using a cloud service to ensure data does not get corrupted is a must. If a drive were to crash, it would be automatically backed up so that gameplay can continue, the user will not lose progress, and neither will the Gaming Room. Linux is a great system for separating all the functions necessary to manage and organize your system. The client will be able to create an account on the server and that is all they should need to worry about. The cloud will take care of the rest of the network.
6. **Security**: Tightening security is a priority, as it should be. The most effective method would be a combination of things, the first being a strong password that is unique, that should go without saying, but it is true. Since Linux is the server, it offers an option to use what it called SSH (Secure Shell) pairing which allows for a higher security level, making it more difficult for hackers to get into the system. Most importantly, keep the server updated at all time, new security threats happen every day, updating is required to stay up to date with them.